



PUC Minas
DIRETORIA DE
EDUCAÇÃO CONTINUADA

Pós Graduação *Lato Sensu*

Bancos de dados não
relacionais

MapReduce e Agregação

Revisando exercício de operações básicas

Document JSON

```
{  
  "_id" : ObjectId("57e344d0e3bdea2a420bd2c9"),  
  "string" : "a",  
  "integer" : 0,  
  "double" : 1.0,  
  "array" : [ 0, 1, 3, 4]  
  "array-string" : [ "d", "c", "b", "a"]  
}
```

Document JSON

```
{  
  "_id" : ObjectId("57e344d0e3bdea2a420bd2c9"),  
  "array-document" : [  
    {  
      "document-name" : "string 1",  
      "document-value" : 0  
    }, {  
      "document-name" : "string 2",  
      "document-value" : 1  
    }  
  ]  
}
```

Document JSON

```
{
  "_id": ObjectId("57e344d0e3bdea2a420bd2c9"),
  "array-document": [
    {
      "document-name": "string 1",
      "document-value": 0
    },
    {
      "document-name": "string 2",
      "document-value": 1
    }
  ]
}
```

Chave ↑ **Valor** ↑

Document JSON

```
{  
  Chave  
  "_id" : ObjectId("57e344d0e3bdea2a420bd2c9"),  
  "array-document" : [  
    {  
      "document-name" : "string 1",  
      "document-value" : 0  
    }, {  
      "document-name" : "string 2",  
      "document-value" : 1  
    }  
  ]  
}
```

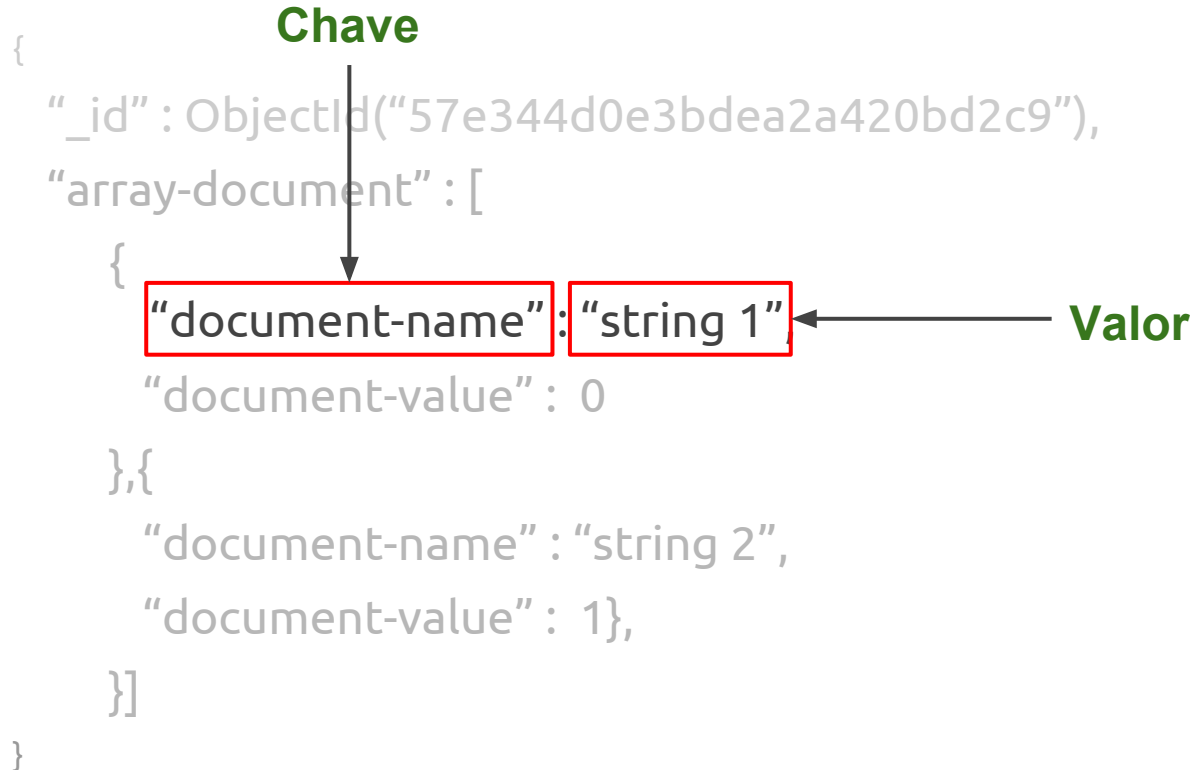
← Valor

Document JSON

Chave

```
{  
  "_id" : ObjectId("57e344d0e3bdea2a420bd2c9"),  
  "array-document" : [  
    {  
      "document-name" : "string 1",  
      "document-value" : 0  
    }, {  
      "document-name" : "string 2",  
      "document-value" : 1  
    }  
  ]  
}
```

Valor



MapReduce

MapReduce

Modelo para processar um grande volume de dados,
dividindo o trabalho em um conjunto de tarefas
independentes.

Tolerante a falhas de processamento e processamento
distribuído.

MapReduce

Map: Funções de mapeamento de dados em operações de chave valor.

Reduce: Percorre os valores que estão associados com a chave e produzir zero ou mais saídas.

MapReduce

Dados

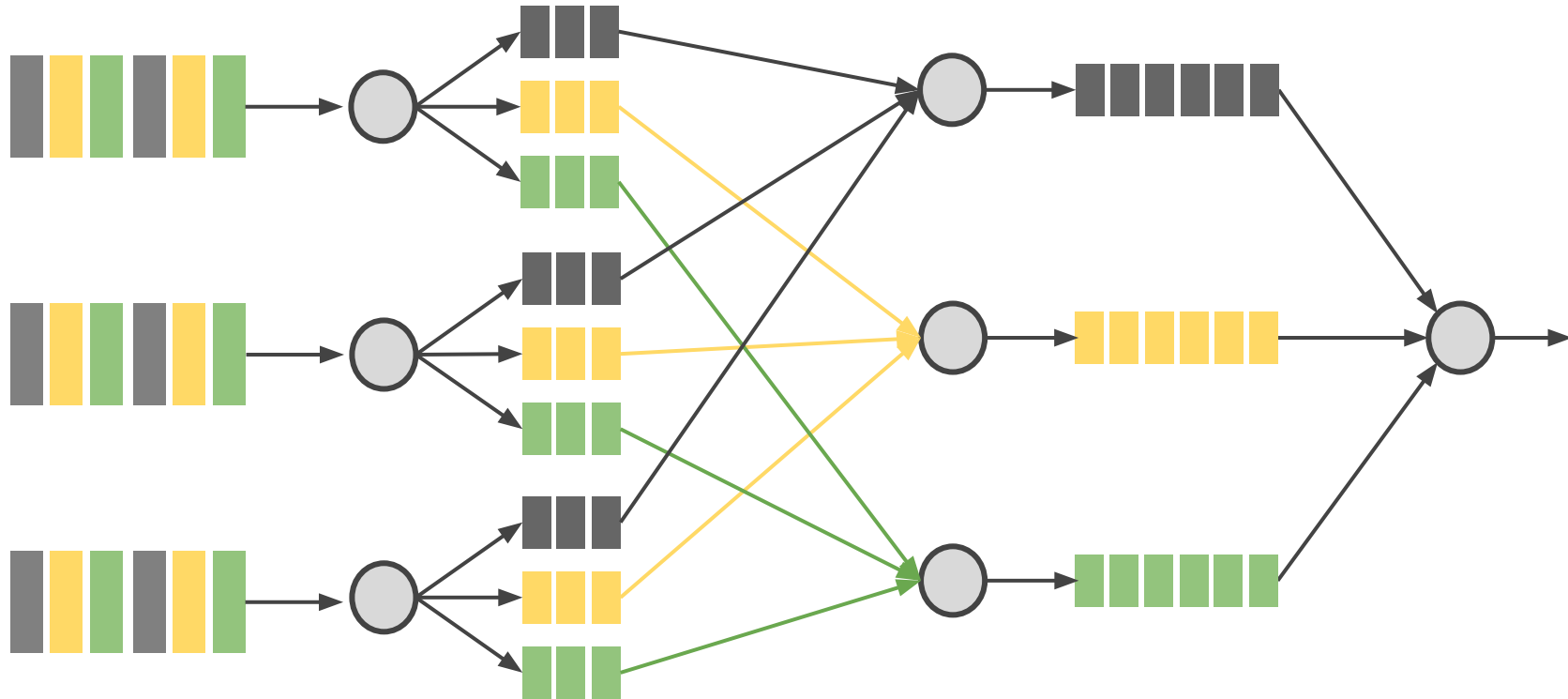


MapReduce

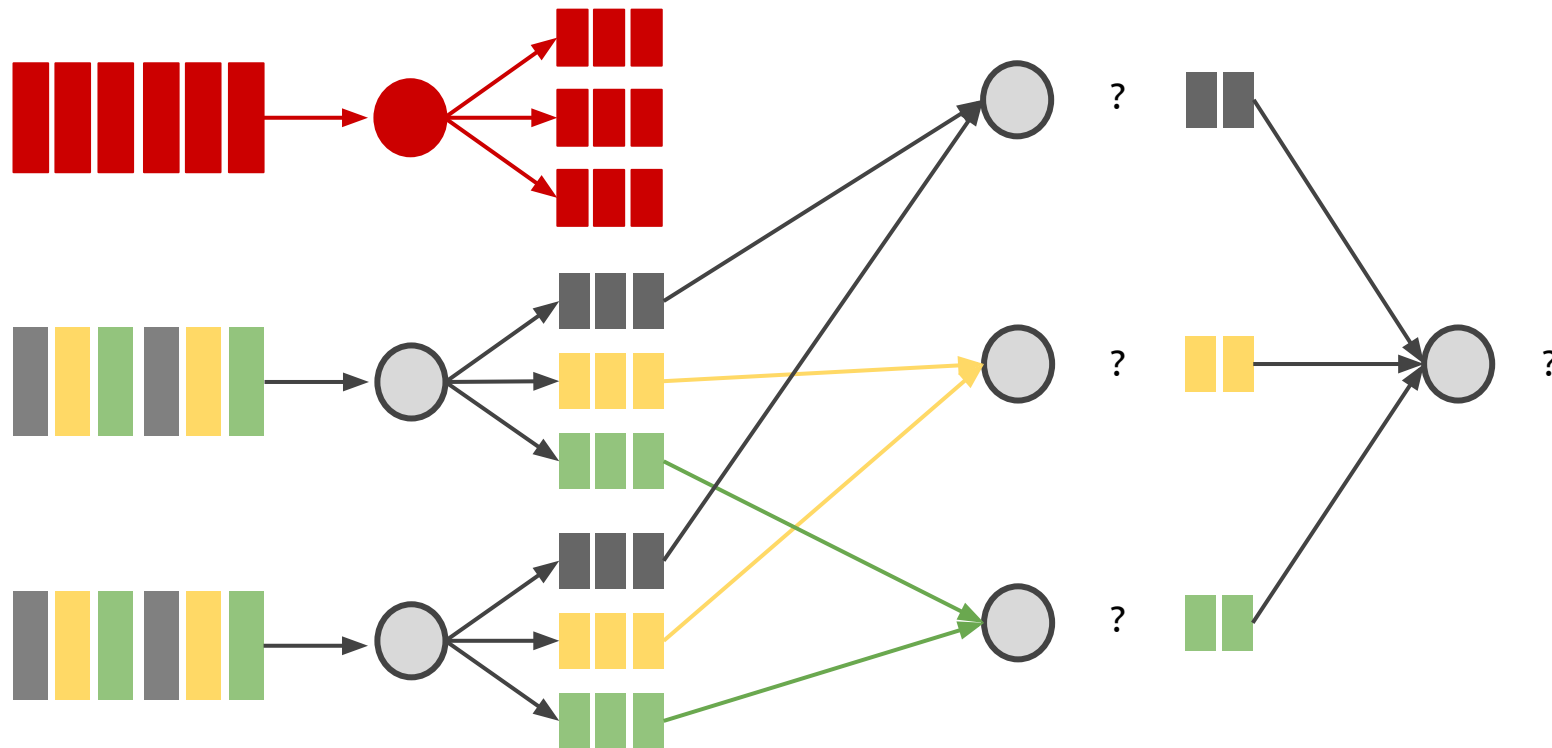
Dados



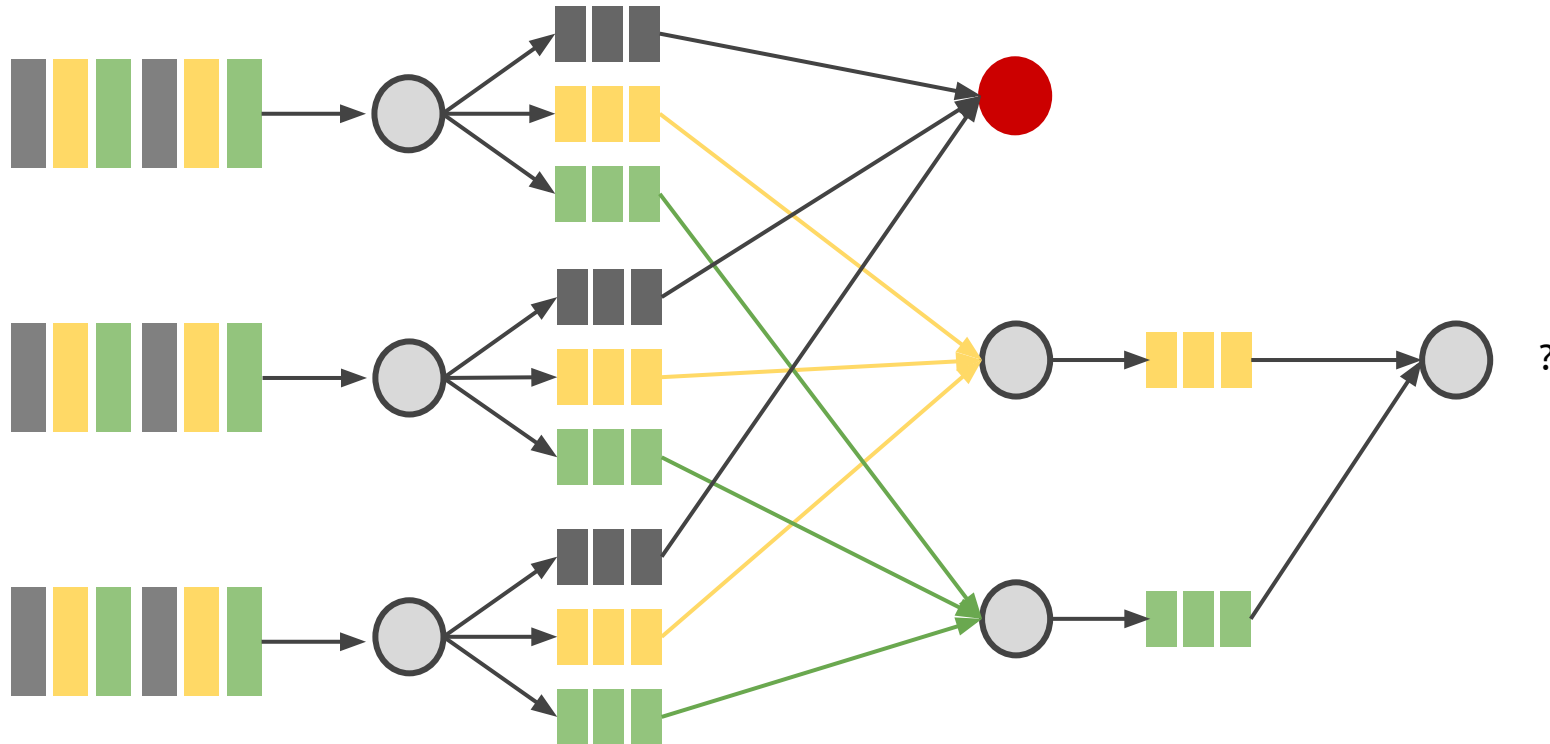
MapReduce



MapReduce



MapReduce



MapReduce - MongoDB

{"cor": "verde"}

{"cor": "verde"}

{"cor": "verde"}

{"cor": "vermelho"}

{"cor": "vermelho"}

{"cor": "cinza"}

```
db.Cores.mapReduce(map, reduce, {query:{}, out: "resultado"})
```


MapReduce - Função Map

```
var map = function(){  
    emit(this.cor, 1);  
}
```

MapReduce - Função Reduce

```
// key = verde e o value = [1,1,1]
```

```
// key = vermelho e o value = [1,1]
```

```
// key = cinza e o value = [1]
```

```
var reduce = function(key, value){  
    return Array.sum(value);  
}
```

MapReduce - Resultado

```
db.resultado.find()
```

```
{ "_id": "verde", value: 3 }
```

```
{ "_id": "vermelho", value: 2 }
```

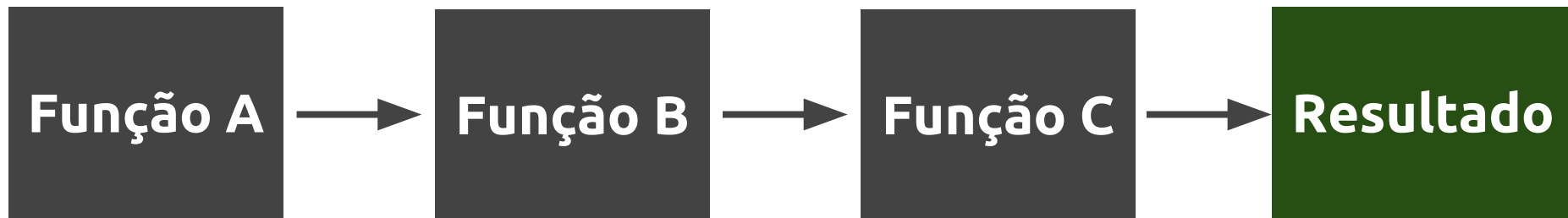
```
{ "_id": "cinza", value: 1 }
```

Aggregate

Agregação (aggregate)

Pipeline de múltiplos estágios para transformação e processamento de documentos.

Agregação - Visão Geral

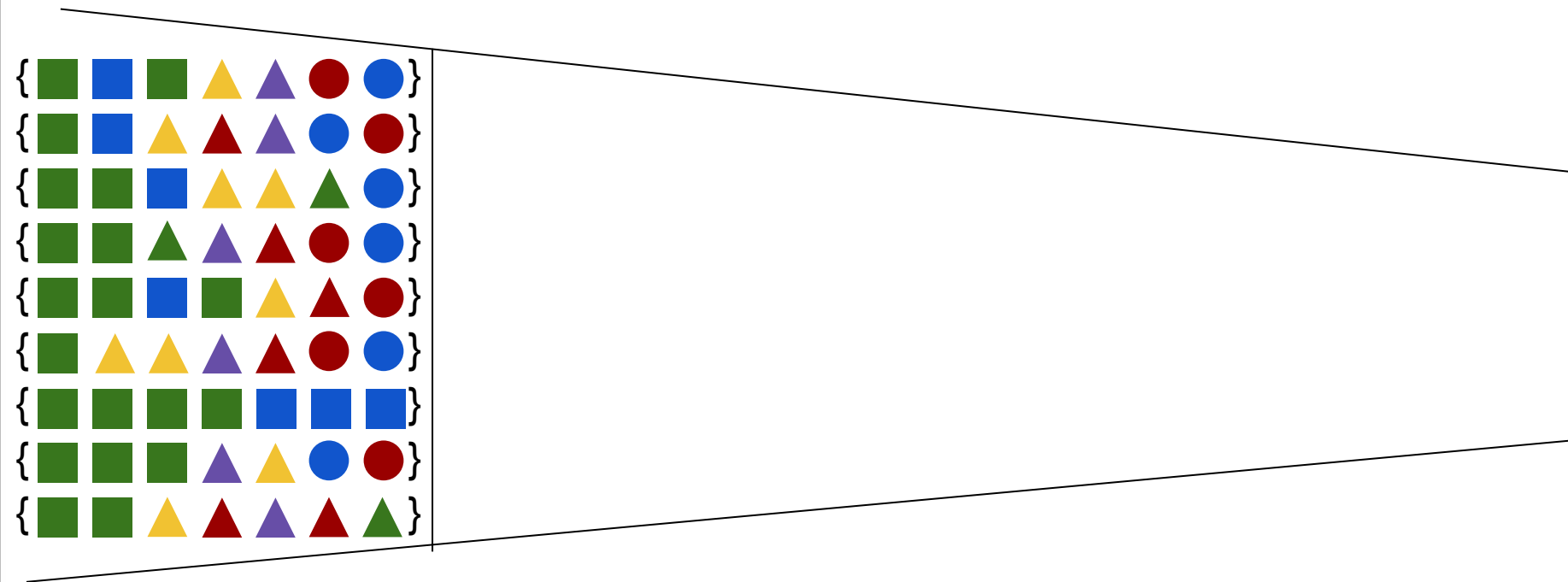


Agregação - Visão Geral

```
db.Aluno.aggregate([ {$match:{...}}, {$project:{...}}, {$group:{...}} ])
```

\$match	Filtra os documentos
\$geoNear	Ordena documentos por proximidade
\$project	Seleciona os campos que passarão para a próxima fase do pipeline
\$unwind	Espande documentos em múltiplos para os arrays
\$group	Agrupamento de documentos
\$sample	Escolhe amostras aleatórias
\$sort	Ordena o pipeline anterior
\$limit	Limita o número de resultados

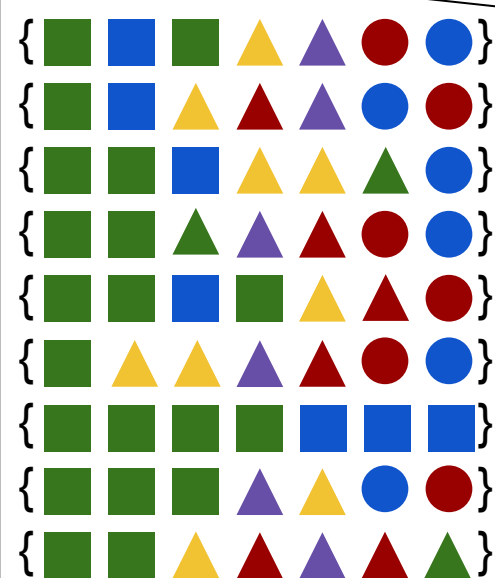
Agregação - Pipeline



Agregação - Pipeline

\$match

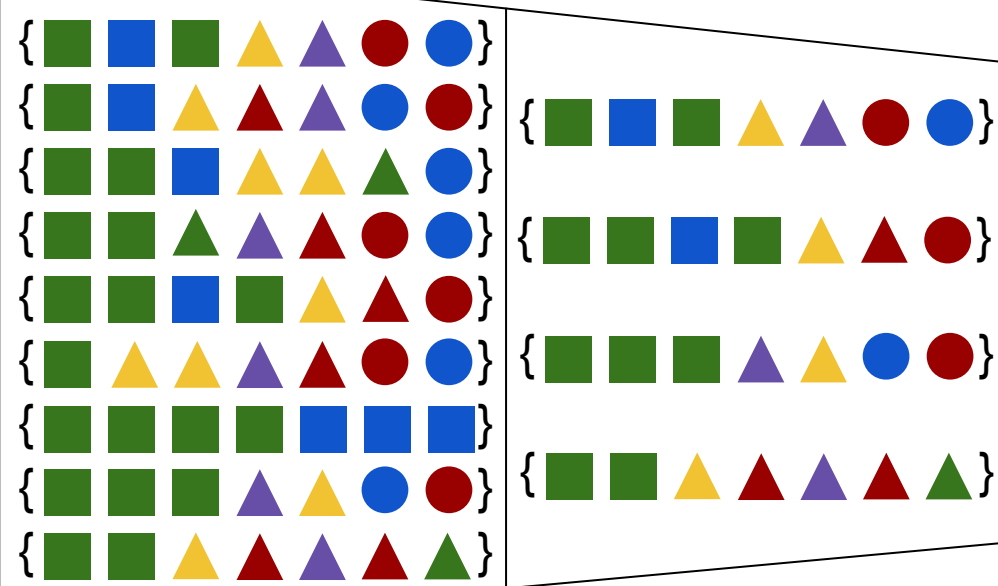
■ \$gte: 2 ▲ :1



Agregação - Pipeline

\$match

■ \$gte: 2 ▲ :1



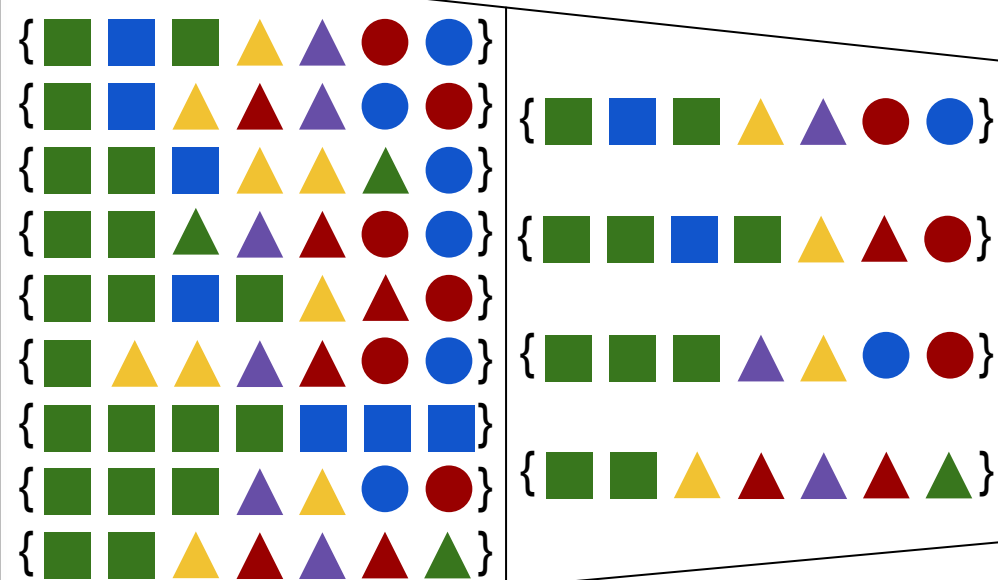
Aggregação - Pipeline

\$match

■ \$gte: 2 ▲ :1

\$project

■ ▲



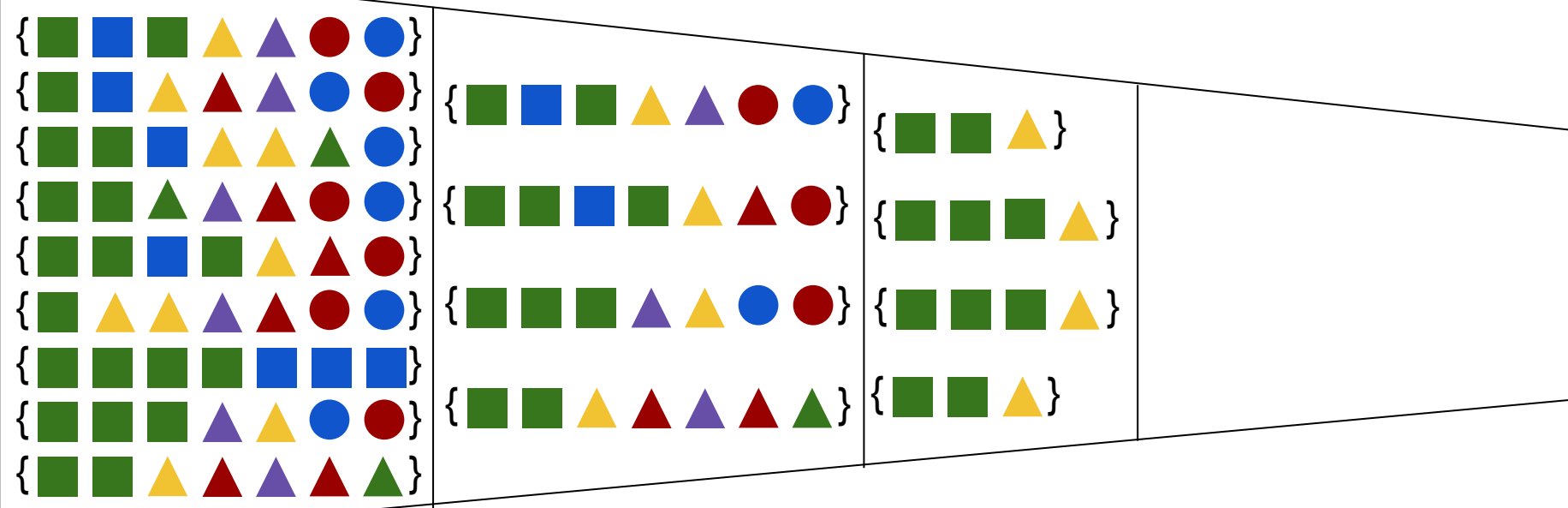
Agregação - Pipeline

\$match

■ \$gte: 2 ▲ :1

\$project

■ ▲



Agregação - Pipeline

\$match

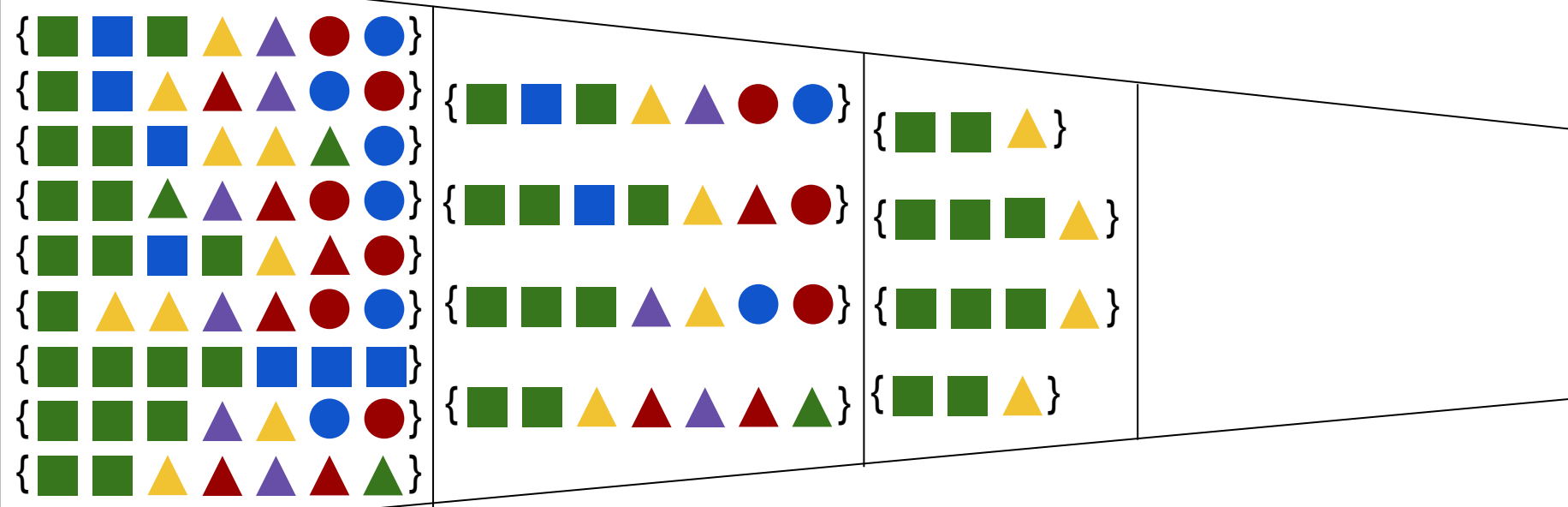
■ \$gte: 2 ▲ :1

\$project

■ ▲

\$group

■



Aggregação - Pipeline

\$match

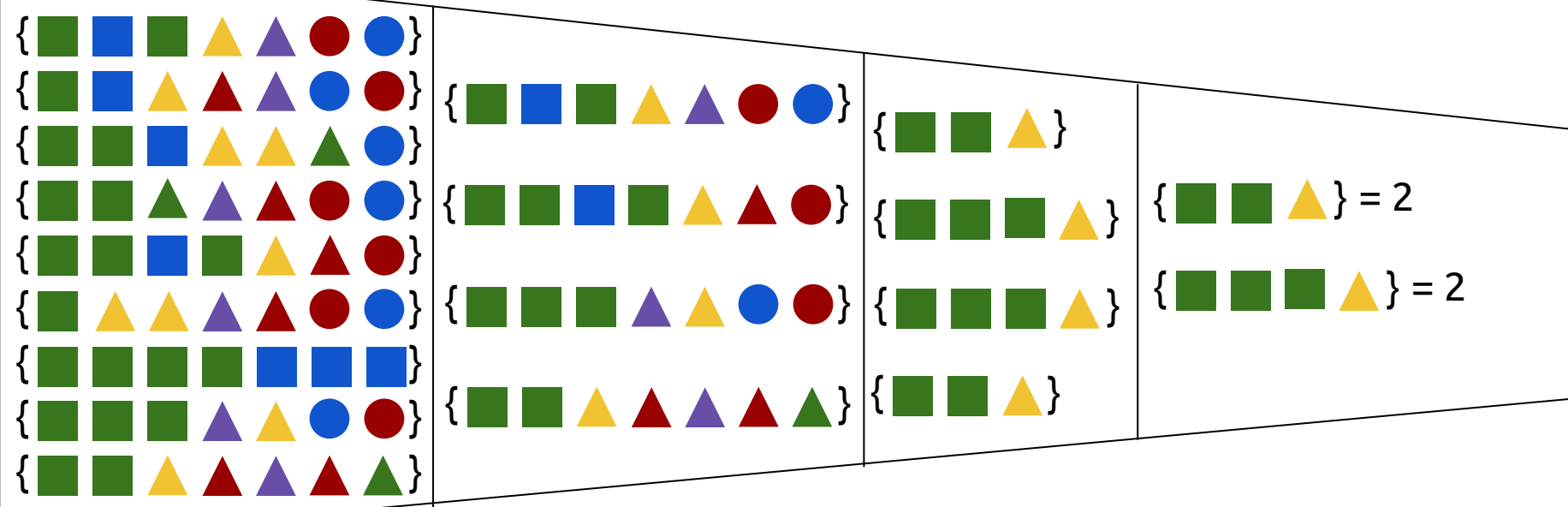
■ \$gte: 2 ▲ :1

\$project

■ ▲

\$group

■



Agregação - \$match

Similiar ao `db.Colecao.find({...})`

`db.Colecao.aggregate({$match:{...}})`

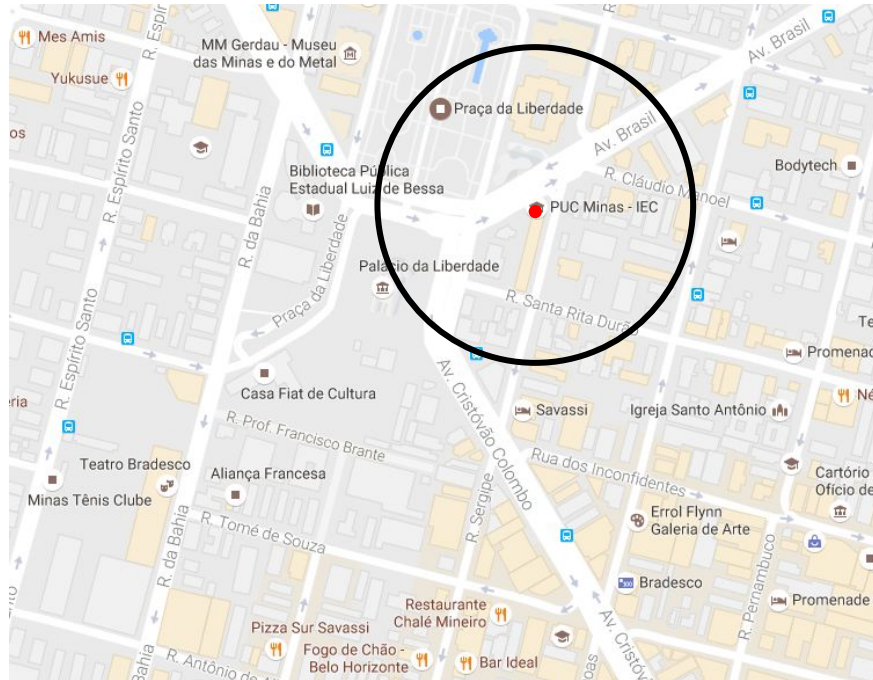


Agregação - \$geoNear

```
{  
  "_id": ObjectId("57e344d0e3bdea2a420bd2c9"),  
  "name": "PUC",  
  "lat" : 0.1232411,  
  "log" : 0.13123123,  
}
```

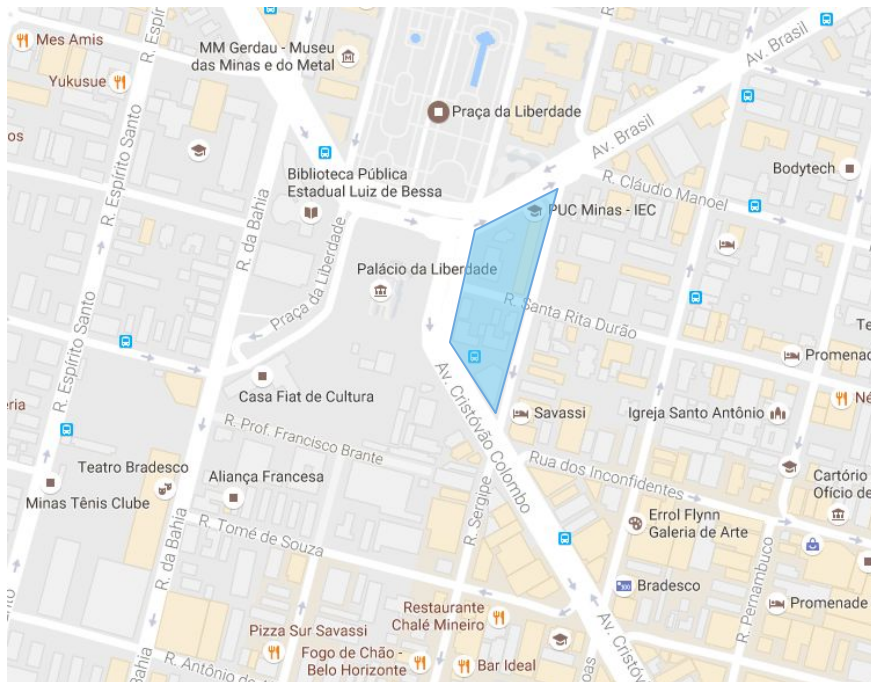

Agregação - \$geoNear

Ponto próximo



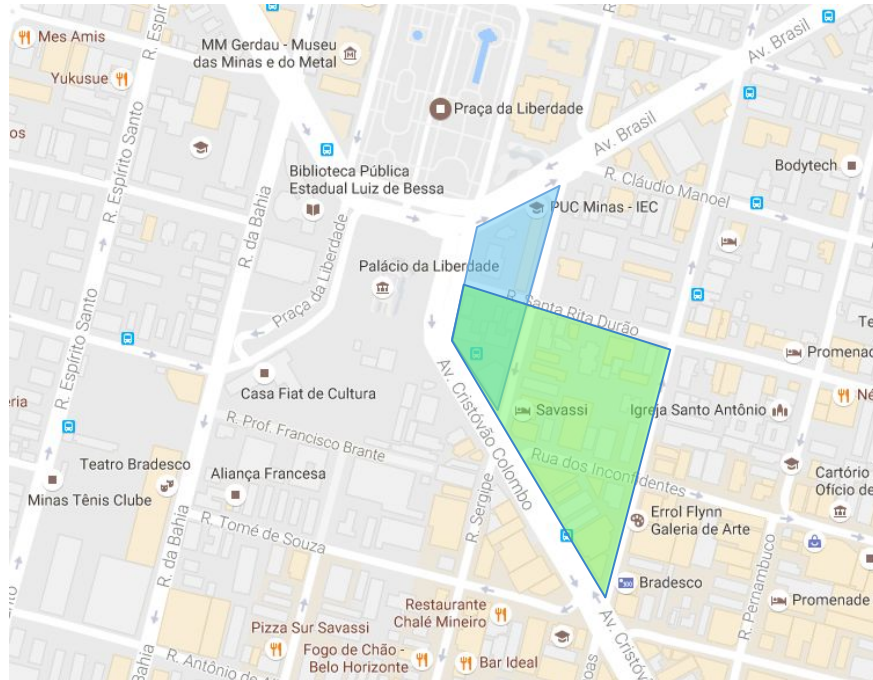
Agregação - \$geoNear

Polígono



Agregação - \$geoNear

Interseção



Agregação - \$unwind

```
{  
  _id: ObjectId("507f191e810c19729de860ea"),  
  name: "Nome",  
  friends: [  
    {"name": "Nome 2"},  
    {"name": "Nome 3"}  
  ]  
}
```

Agregação - \$unwind

```
db.Colecao.aggregate({$unwind:"$friends"})
```

```
{_id: ObjectId("507....9de860ea"), name: "Nome", friends: {"name": "Nome 2"}}
```

```
{_id: ObjectId("507...9de860ea"), name: "Nome", friends: {"name": "Nome 3"}}
```

Agregação - \$group

```
{  
  _id: ObjectId("507f191e810c19729de860ea"),  
  name: "Nome",  
  birthday: ISODate("2016-11-10T19:13:00Z"),  
  informations: {  
    city: "Belo Horizonte",  
    state: "Minas Gerais"  
  },  
  friends: [  
    {"name": "Nome 2"},  
    {"name": "Nome 3"}  
  ]  
}
```

Agregação - \$group

```
{
  { $group:
    {
      "_id":
      "$informations.city",
      "total": {
        $sum: 1
      }
    }
  }
}
```

```
{
  _id: ObjectId("507f191e810c19729de860ea"),
  name: "Nome",
  birthday: ISODate("2016-11-10T19:13:00Z"),
  informations: {
    city: "Belo Horizonte",
    state: "Minas Gerais"
  },
  friends: [
    {"name": "Nome 2"},
    {"name": "Nome 3"}
  ]
}
```

Agregação - \$group

Uso de \$ para acessar atributos do pipeline

```
{ $group:
```

```
{
```

"_id":

"\$informations.city",

"total": {

\$sum: 1

```
}
```

```
}
```

```
}
```

```
_id: ObjectId("507f191e810c19729de860ea"),
```

```
name: "Nome",
```

```
birthday: ISODate("2016-11-10T19:13:00Z"),
```

```
informations: {
```

```
city: "Belo Horizonte",
```

```
state: "Minas Gerais"
```

```
},
```

```
friends: [
```

```
{ "name": "Nome 2" },
```

```
{ "name": "Nome 3" }
```

```
]
```

```
}
```


Agregação - \$group

```
{  
  $group:  
  {  
    "_id":  
    "$informations.city",  
    "total": {  
      $sum: 1  
    }  
  }  
}
```

Operadores

\$sum, \$mul, \$max, \$min, \$avg, \$first,
\$last, \$push, \$addToSet, \$stdDevPop,
\$stdDevSamp

Agregação - \$group

```
{
```

```
  $group:
```

```
  {
```

```
    "_id":
```

```
    "$informations.city",
```

```
    "total": {
```

```
      $sum: 1
```

```
    }
```

```
  }
```

```
}
```

Resultado

```
{
```

```
  "_id": "Belo Horizonte",
```

```
  "total": 1
```

```
}
```

Preparando para os exercícios

Ligar a máquina virtual - NoSQL

1 - Abra **dois** terminais

2 - No primeiro terminal ligue o mongod com o comando:

```
cd ~/Aulas/mongodb; ./bin/mongod --dbpath=../dados
```

3 - No segundo terminal importe os dados para essa aula executando o comando:

```
cd ~/Aulas/mongodb; ./bin/mongoimport -d nosqlclass -c Vocabulary ../nosql-class/aula2/Vocabulary.json
```

4 - Após importar abra o mongo shell com o comando:

```
cd ~/Aulas/mongodb; ./bin/mongo
```

Exercício 1

Na coleção Vocabulary

- A) Utilizando as funções de mapReduce do mongo, conte o número de palavras que terminam em ar, er, ir, or, ur.
- B) Utilizando as funções de mapReduce do mongo, conte o total de cada caracter existente no vocabulario. Por exemplo:
aula -> a:2, u:1, l:1

Para facilitar você pode escrever o código em um documento de texto e importa-lo usando a função load() do mongo shell

Ex.: > load("/home/nosql/Aulas/nosql-class/aula3/ex1.js")

Exercício 2

Utilizando a função de agregação contar quantos itens cujo o campo total seja maior do que 1000, agrupando-os por tipo, (campo type) e exiba o resultado em ordem crescente.

Funções úteis de JavaScript

Tamanho de uma string

nome.length

Substring javascript

nome.substring(nome.length-2,nome.length)

Print do mongoshell

print('hello')